Applying Asynchronous Circuits in Contactless Smart Cards

Joep Kessels (Philips Research)*
Torsten Kramer (MAZ Hamburg)§
Gerrit den Besten, Ad Peeters (Philips Research)
Volker Timm (Philips Semiconductors)

Abstract

We have designed an asynchronous chip for contactless smart cards. Asynchronous circuits have two power properties that make them very suitable for contactless devices: low average power and small current peaks. The fact that asynchronous circuits operate over a wide range of the supply voltage, while automatically adapting their speed, has been used to obtain a circuit that is very resilient to voltage drops while giving maximum performance for the power being received.

The asynchronous circuit has been built, tested and evaluated and the results were so convincing that, based on the circuits presented, a product is being designed.

Keywords: *low-power asynchronous circuits, smart cards, contactless devices, DES cryptography.*

1. Introduction

Since their introduction in the eighties, smart cards have been used in a continuously growing number of applications, such as banking, telephony (telephone and SIM cards), access control (Pay-TV), health-care, tickets for public transport, electronic signatures and identification. Currently, most cards have contacts and, for that reason, need to be inserted into a reader. For applications in which the fast handling of transactions is important, so-called *contactless smart cards* have been introduced requiring only close proximity to a reader (typically several centimeters). The chip on such a card must be extremely power efficient, since it is powered by electromagnetic radiation.

Asynchronous CMOS circuits have the potential for very low power consumption, since they only dissipate when and where active. However, asynchronous circuits are difficult to design at the level of gates and registers. Therefore the high-level design language Tangram was defined [10] and a so-called silicon compiler has been implemented that translates Tangram programs into asynchronous circuits. Tangram is a conventional programming language, like C or Pascal, extended to include constructs for expressing concurrency and communication in a way similar to the language CSP [2]. Similar approaches have been proposed in [1, 3].

The Tangram compiler generates a special class of asynchronous circuits called handshake circuits [8, 6]. Handshake circuits are constructed from a set of about 30 basic components that use 4-phase single-rail handshake signalling for communication.

Several chips have been designed in Tangram [9, 11] and if we compare these chips to existing clocked implementations, then the asynchronous versions are generally about 20% larger in area and consume about 25% of the power.

In order to find out what advantages asynchronous circuits have to offer in contactless smart cards, we have designed the digital circuitry of a smart card chip as well as the analog power regulator. In this paper, we indicate which properties of asynchronous circuits have been exploited and we present the results. The paper is organised as follows. Section 2 provides some background to contactless smart cards and identifies the power characteristics in which contactless devices differ from battery-powered ones. Section 3 elaborates on the differences in power behaviour between synchronous and asynchronous circuits and indicates how these differences can be exploited in contactless devices. The next two sections present the design, section 4 discussing the digital circuit and section 5 the power supply unit. In section 6 we conclude with the results and a summary of the merits of this asynchronous design.

2. Contactless smart cards

Fig. 1 shows the functional parts of a contactless smart card consisting of a VLSI circuit (in the dotted box) and an external coil. The tuned circuit formed by the coil and capacitor C_0 is used for three purposes:

*Joep.Kessels@philips.com
§kramer@maz-dd.de

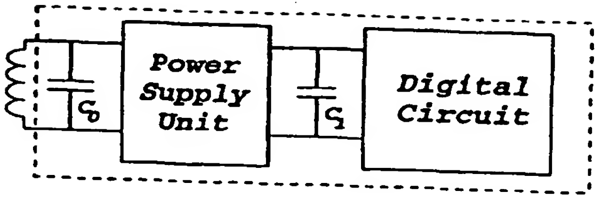

Figure 1. Contactless smart card

- receiving power;
- receiving the clock frequency (equal to the carrier frequency); and
- supporting the communication.

The circuit proper consists of a Power Supply Unit and a digital circuit with a buffer capacitor (C_1) for storing energy.

Contactless smart cards have a number of advantages when compared to contacted ones: they are convenient and fast to use, insensitive to dirt and grease and, since their readers have no slots, they are less amenable to vandalism.

Since contactless smart card chips receive only a few mW of power, power efficiency is very important. Although low power is also important in battery-powered devices, there are two crucial differences between both kinds of devices.

1. To maximize the battery life-time in battery-powered devices, one should minimize the *average* power consumption. In contactless devices, however, one should in addition minimize the *peak* power, since the peaks must be kept below a certain level, depending on the incoming power as well as the buffer capacitor.
2. The supply voltage is nearly constant in battery powered devices, whereas in contactless ones it may vary ver time during a transaction due to fluctuations in both the incoming and the consumed power.

Several standards for contactless smart cards, currently, exist. The main standard, however, is Mifare [4], which has currently sold about 70 million cards. Mifare is a proximity card (it can be used up to 10 cm) supporting two way communication. The carrier frequency is 13.56 Mhz and the communication rate is 106 Kbit/sec. Performance is important, since the transaction time must be less than 200 msec. One of the first companies to deploy Mifare technology *en masse* was the Seoul Bus Association, which currently has 6 million bus cards in use, generating 80 million transactions per month.

In the bullets below, we give some facts about conventional synchronous chips for contactless smart cards, which, as we will see later, offer opportunities for improvement by using asynchronous circuits.

- A synchronous circuit runs at a fixed speed dictated by the clock, despite the fact that both the incoming and the effectively consumed power vary over time. Synchronous circuits must, therefore, be designed so as to allow the most power-hungry operations to be performed when minimum power is coming in. Consequently, if too much power is being received, that superfluous power is thrown away. If, on the other hand, too little power is being received, the transaction must be canceled. If too little power comes in, the supply voltage drops making the circuit slower and, as soon as the circuit has become too slow to meet the time requirements set by the clock, the transaction must be canceled. For this reason contactless smart card chips contain subcircuits that detect when the voltage drops below a certain threshold and then abort the transaction.
- Currently, the performance of the microcontroller in a contactless smart card chip is usually not limited by the speed of the circuit, but by the RF-power being received.
- A synchronous circuit requires a buffer capacitor of several nanoFarads and the area needed for such a capacitor is of the same order of magnitude as the area needed for the microcontroller.
- The communication from the smart card to the reader is based on modulating the load, which implies that normal functional load fluctuations may interfere with the communication.

3. Differences between synchronous and asynchronous circuits

When the asynchronous circuits generated by the Tangram compiler are compared to synchronous ones, three differences stand out, leading to four attractive properties of asynchronous circuits.

1. The subcircuits in a synchronous circuit are clock-driven, whereas they are demand-driven in an asynchronous one. This means that the subcircuits in an asynchronous circuit are only active when and where needed. Asynchronous circuits will therefore generally dissipate less power than synchronous ones.
2. The operations in a synchronous circuit are synchronized by a central clock, whereas they are synchro-

nized by distributed handshakes in an asynchronous circuit. Therefore

a) a synchronous circuit shows large current peaks at the clock edges, whereas the power consumption of an asynchronous circuit is more uniformly distributed over time;

b) the strict periodicity of the clock in a synchronous circuit leads to higher harmonics in the frequency spectrum, which are absent in the spectrum of an asynchronous design.

3. Synchronous circuits use an external time reference, whereas asynchronous circuits are self-timed. This means that asynchronous circuits operate over a wide range of the supply voltage (for instance, from 1 up to 3.3 V) while automatically adapting their speed.

Property 2.b was the main reason for Philips Semiconductors to design a family of asynchronous pager chips[1]. However, it is the other properties that can be used advantageously in contactless smart card chips. Since it is the peak power that matters for contactless applications, both advantage 1 and 2.a are relevant, and property 3 can be used to cope with the fluctuating supply voltage.

4. The digital circuit

We have built the digital circuit shown in fig. 2 that consists of:

- an 80C51 micro-controller;
- three kinds of low-power memory, of which the sizes and accesstimes are given in Table 1 (64 bytes can be written simultaneously in one write access to the EEPROM);
- two encryption co-processors:
  - an RSA converter [7] for public key conversions and
  - a triple DES converter [5] for private key conversions;
- a UART for the external communication.

The EEPROM contains program parts as well as data such as encryption keys and e-money. Both the ROM and the RAM are equipped with matching delay lines and for the EEPROM we designed a similar function based on a counter. These delay lines have been used to provide all three memories with a handshake interface, which made it extremely easy to deal with the differences in access time as well as variations in both temperature and supply voltage.

The circuit is meant to be used in a so-called dual interface card, which is a card with both a contacted and a contactless interface. Apart from the RSA converter, which will not be used in contactless operation, all circuits are asynchronous. In contactless operation, the average supply voltage will be about 2 V. The simulations, however, are done at 3.3 V, which is the voltage at which the library has been characterized.

[1]http://www.semiconductors.philips.com/pip/PCA500711

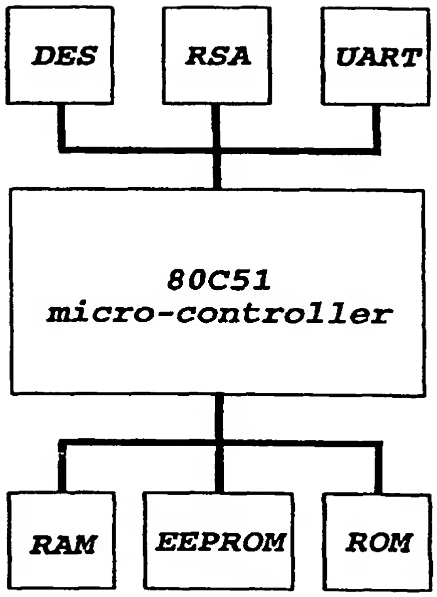

Figure 2. Global design digital circuit

Memory type	Size [kbyte]	Access time [ns]	
		read	write
RAM	2	10	10
ROM	38	30	30
EEPROM	32	180	4,000

Table 1. Memory sizes and access times

4.1. The 80C51 micro-controller

The 80C51 micro-controller is a modified version of the one described in [11]. The four most important modifications are described below.

To deal with the slow memories a *prefetch unit* has been included in the 80C51 architecture. At 3.3 V the average instruction execution time in free-running mode is about 100 ns provided it takes no time to fetch the code from memory. If, however, code is fetched from the EEPROM and the microcontroller would have to wait during such read accesses, the performance would be drastically reduced, since most

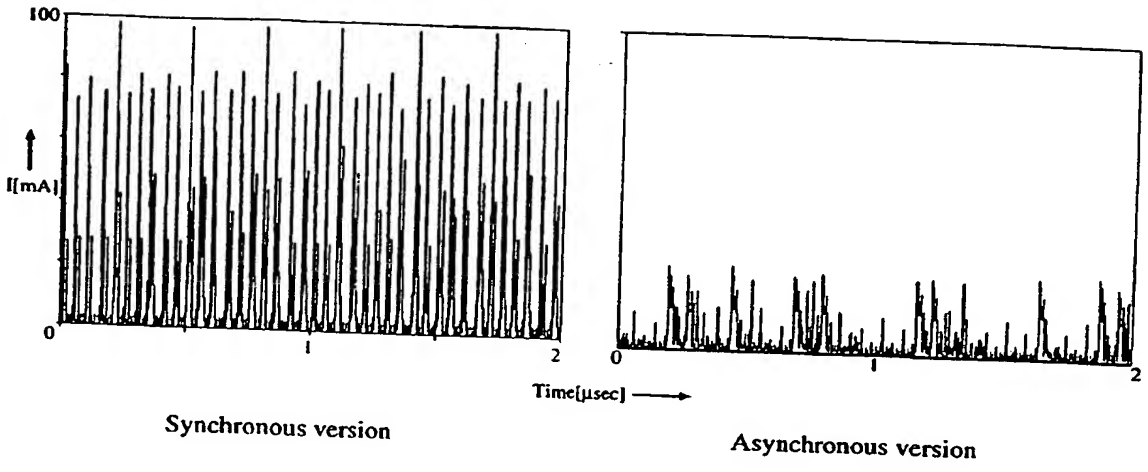

Figure 3. Current peaks of 80C51 micro-controller

instructions are one or two byte long, taking 180 or 360 ns to fetch. To avoid this performance degradation a form of *instruction prefetching* has been introduced in which a process running concurrently to the 80C51 core is fetching code bytes as long as a two byte fifo is not full. The prefetch unit gives an increase in performance of about 30%. A simplified version of the prefetch unit is described in the next subsection.

We also introduced *early write completion*, which means that the micro-controller continues execution as soon as it has issued a write access. It has been introduced to prevent the microcontroller from waiting during the 4 msec it takes to do a write access to the EEPROM (for instance to change the e-money), but also to speed up the write accesses to the RAM. To exploit this feature when doing a write access to the EEPROM, the corresponding code must be in the ROM.

The controller has been provided with an *immediate halt* input signal by which its execution can be halted within a short time. This provision is necessary to deal with the fact that the information, which the reader sends to the card, is coded by suppressing the carrier during periods of 3 μsec. Since the card does not receive any power during these periods, the controller has to be halted immediately (only some basic functions continue to operate). In the synchronous design this halting function came naturally, since the clock would stop during these periods.

We have introduced a so-called *quasi synchronous* mode, in which the micro-controller is, at instruction level, fully timing compatible with its synchronous counterpart. In this mode, the asynchronous micro-controller waits after each instruction until the number of clock ticks required by a synchronous version have elapsed. This mode is necessary when time dependent functions are designed in software. Since this mode is under software control, the micro-controller can easily switch modes depending on the function it is executing. This feature was also of great help to demonstrate the *guaranteed performance*, which is the maximum clock rate at which each instruction terminates within the given number of clock ticks. For most programs, the *free-running performance* is about twice as high as the guaranteed performance.

The micro-controller nicely demonstrates the three properties of asynchronous circuits that we want to exploit in the design of the smart card chip. We could compare the asynchronous version with a synchronous one, where the synchronous one gives a comparable performance.

- The average power consumption of the asynchronous 80C51 is about three times lower than the power consumption of its synchronous counterpart when delivering the same performance at the same supply voltage.
- Fig. 3 shows the current peaks of both the synchronous and the asynchronous 80C51 at 3.3 V, where the asynchronous version is running in quasi synchronous mode, giving a performance that is 2.5 times higher than the synchronous one (the synchronous one runs at 10 MHz and the asynchronous one at 25 MHz). Despite the fact that the figure does not give a fair impression of the average power being consumed, it clearly shows that the current peaks of the asynchronous 80C51 are about five times smaller than those of the synchronous one.
- The performance adaptation property of asynchronous circuits is demonstrated in fig. 4, which shows the free-running performance of the micro-controller, when executing code from ROM, as a function of the supply

voltage. As could be expected, the performance depends linearly on the supply voltage. When the supply voltage goes up from 1.5 to 3.3 V, the performance increases from 3 to 8.7 MIPS (about a factor 3). Since the ROM containing the program does not function properly when the supply voltage is below 1.5 V, we could not measure the performance for lower values. We observed, however, that the DES co-processor, which does not need a memory, still functions correctly at a supply voltage level as low as 0.5 V.

The figure also shows the supply current as a function of the supply voltage. Note that the current increases in this range from 0.7 to 6 mA (about a factor 9). Since in CMOS circuits the current is the product of the transition rate (performance) and the charge being switched per transition (both of which depend linearly on the supply voltage), the current increases with the square f the voltage. From this it follows that the power, being the product of the current and the voltage, goes up with the cube of the voltage.

From this data one can compute the third curve showing the energy needed to execute an instruction, which increases with the square of the supply voltage from 0.35 to 2.25 nJ.

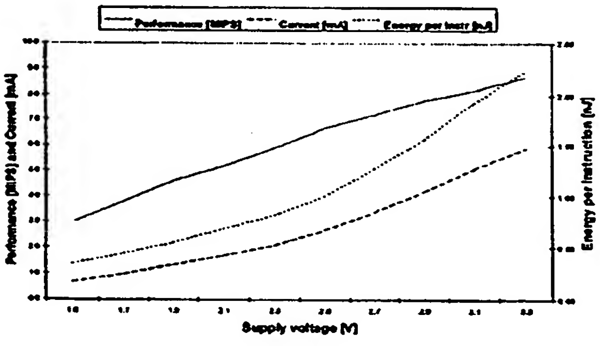

Figure 4. Performance adaptation of asynchronous 80C51

4.2. The prefetch unit

Fig. 5 gives the Tangram code of a simplified version of the prefetch unit. The prefetch unit communicates with the 80C51 core through two channels: it receives the address from which to start fetching code bytes via channel *StartAddress* and it subsequently sends these bytes through channel *CodeByte*. Since the prefetch unit plays in both communications the passive role, it can probe each channel to see whether the core has started a communication through that channel. The state of the prefetch unit consists of program counter *pc*, and a two-place buffer, which is implemented by means of an array *Buffer*, an integer *count*, and two one-bit pointers *getptr* and *putptr*.

```
forever
do  sel probe(StartAddress)
          then StartAddress?pc || putptr := getptr ||
               count := 0 || AbortMemAcc()
    or probe(CodeByte) ∧ (count > 0)
          then CodeByte!Buffer[getptr];
               ( getptr := next(getptr)
               || count := count − 1
               )
    or MemAck
          then Buffer[putptr] := MemData;
               ( putptr := next(putptr)
               || count := count + 1
               || pc := pc + 1 || CompleteMemAcc()
               )
    les;
    if (count < 2) ∧ (¬MemReq)
    then MemReq := true
    fi
od
```

Figure 5. Tangram code of simplified version of prefetch unit

The prefetch unit executes an infinite loop and in each step it first executes a selection command, in which it can select among three so-called guarded commands. Each guarded command is of the form "*guard* **then** *command*", the guarded commands are separated by the key word "**or**" and the list is enclosed by the bracket pair "**sel**" and "**les**". A command is said to be *enabled* if the corresponding guard holds. Executing a selection command implies waiting until at least one of the commands is enabled, then selecting, in an arbitrated choice, such a command and executing it. In the first guarded command, channel *StartAddress* is probed to find out whether the core is sending a new start address. In that case, *pc* is set to the address received, the buffer is flushed and a possible outstanding memory access is aborted (by resetting both *MemReq* and the delay counter). All four subcommands are executed concurrently ("$A||B$" means execute A and B concurrently, whereas "$A; B$" means execute A and B sequentially). The second guarded command takes care of sending the next program byte via channel *CodeByte* to the core if the core is ready to receive that byte and the buffer is not empty. The third guarded command gets enabled if *MemAck* goes high indicating that the data signals in a read access are valid. In that case the value read from memory is put in the buffer after which the memory handshake is completed.

After each event a next memory access is started in case the buffer is not full while no memory access is being performed (since $(count < 2) \vee (\neg MemReq)$ holds, the last command in the loop step can be simplified to $MemReq := (count < 2)$).

Note that the value $(pc - count)$ is equal to the program counter in the core, since it is set to the destination address in case of a jump, increased by 1 if a code byte is transferred to the core, and kept invariant if a code byte is read from memory. Therefore the core does not need to hold the program counter and instead, when the information is needed for a relative branch, it can retrieve the counter value from the prefetch unit. Clearly, this feature requires an extension of the Tangram code shown in fig. 5.

4.3. The DES co-processor

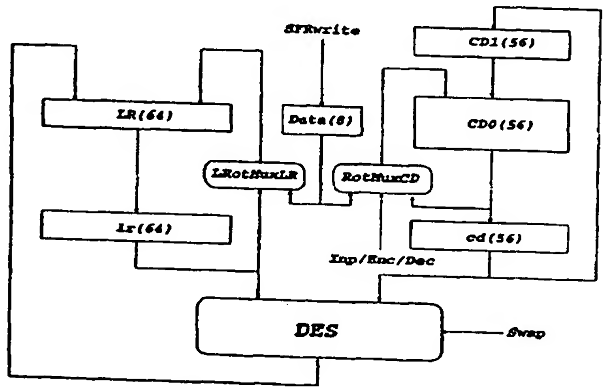

Figure 6. DES co-processor architecture

A transaction may need up to ten single DES conversions, where each conversion takes about 10 msec if it is executed in software. Therefore a hardware solution is needed, since these conversions would otherwise consume about half of the transaction time.

Fig. 6 shows the data path of the DES co-processor. The processor supports both single and triple-DES conversions and, for the latter type of conversion, contains two keys: a foreground and a background key. Single-DES conversions use the foreground key, whereas triple-DES conversions use the foreground key for the first and third conversion and the background key for the second one. The foreground key is stored in register *CD0* consisting of 56 flipflops (the DES key size is 56 bits), whereas the background key resides in variable *CD1* consisting of 56 latches. The text value resides in variable *LR* consisting of 64 latches (DES words contain 64 bits).

A single-DES conversion consists of 16 steps and, in each step, both the key is permuted and a new text value is computed from both the old one and the key. In order to have the key return to its original value at the end of a conversion, the key makes two basic permutations at 12 steps and only one at the remaining 4, where 28 basic permutations are needed for a complete cycle. The permutations are performed in flipflop register *CD0*.

Most of the area is taken by the combinational circuit called *DES*. Since this circuit is also dominant in power dissipation, one should minimize the number of transitions at its inputs. For this purpose, we have introduced two latch registers: *cd* for the key and *lr* for the text. If two basic permutations are done in one step, *cd* hides the effect of the first one from combinational circuit *DES*. In addition, all inputs of combinational circuit *DES* change only once in each step by loading the two registers *lr* and *cd* simultaneously and then storing the result in register *LR* as described by the following piece of Tangram text.

$$(lr := LR \;\|\; cd := CD0);\;\; LR := DES(lr, cd).$$

Therefore, latch register *lr* also serves as a kind of slave register. Latch register *cd* also serves a functional purpose, since the two keys are swapped by executing the following three transfers:

$$cd := CD0;\;\; CD0 := CD1;\;\; CD1 := cd.$$

The size of the DES co-processor is 3,250 gate equivalents, of which 57% is taken by the combinational logic and 35% by latches and flipflops. Consequently, the overhead in area due to the asynchronous design style (delay matching and C-elements) is marginal at 8%. At 3.3 V, a single-DES conversion takes 1.25 μs and 12 nJ.

Fig. 7 shows the simulated current of the DES co-processor at 3.3 V (the micro-controller is active before and after the DES computation). The real current peaks will be much smaller due to a lower supply voltage (the DES processor functions properly at a supply voltage as low as 0.5 V) as well as the buffer capacitor (the resolution in the simulation is 1 ns).

The conversion time, of a few microseconds, is so small that we used the handshaking mechanism to obtain the synchronization between the micro-controller and the co-processor. After starting the co-processor, the micro-controller can continue executing instructions and only when reading the result, will it be held up in a handshake until the result is available. Note that a synchronous design would require a form of busy waiting.

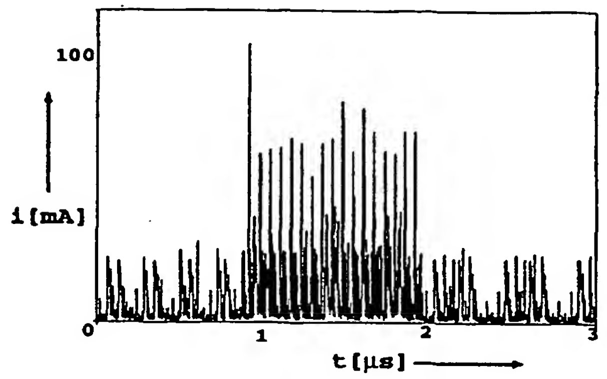

Figure 7. Current DES co-processor at 3.3 V

4.4. Results

Fig. 8 shows the layout of the chip, which is in a five-layer metal, 0.35 μm technology and has a size of 4.52 $\times$ 4.16 $\approx$ 18 mm^2, including the bonding pads. Many bonding pads are only included for measurement and evaluation purposes. A production chip only needs about 10 bonding pads.

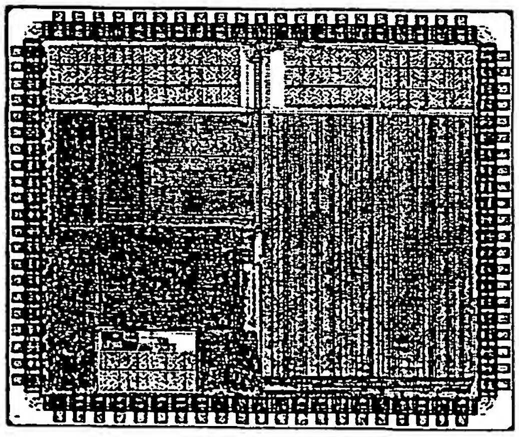

Figure 8. Layout of smart card chip

The two horizontal blocks on top form the buffer capacitor (in a production chip, such a capacitor would only require about one quarter of the area size). The memories are on the next row, from left to right: two RAMs, one ROM and the EEPROM, which is the large block to the right. The asynchronous circuit is located in the lower left quadrant, near the center.

Table 2 gives the area f the blocks constituting the *contactless digital circuit*, which is the asynchronous circuit together with the memories. The other modules are either synchronous or analog circuits, where the synchronous modules are not used in contactless operation. From this table it follows that the asynchronous logic takes only 12% of the total contactless digital circuit.

Block	Area [mm^2]
RAM	1.2
ROM	1.0
EEPROM	5.6
Async. circ.	1.1
Total	8.9

Table 2. Area contactless digital circuit blocks

The sizes of the different asynchronous modules are given in Table 3. In the standard cell library used, a gate equivalent (GE) is 54 μm^2 with a typical layout density of 17,500 gates per mm^2.

Module	Area [GE]
CPU	7,800
Pref. Unit	700
DES	3,250
UART	2,040
Interfaces	3,680
Timer	1,080
Total	18,550

Table 3. Area asynchronous modules

Table 4 shows the power dissipation of the digital circuit blocks when the controller is executing code from ROM (being the normal situation).

Block	Power [%]
Core	56
ROM	27
RAM	17

Table 4. Power of contactless digital circuit

Table 5 shows the effects on power and area of an asynchronous design at two different levels. The asynchronous circuit proper gives a reduction in power dissipation of about 70% for 18% additional area. At the level of the contactless digital circuit, however, we obtain a power reduction of 60% for only 2% additional area. Note that this analysis does not include the synchronous RSA converter and the analog circuits needed in a production chip, such as for instance the buffer capacitor and the power supply unit. Therefore at chip level the relative reduction in power dissipation will be about the same, whereas the overhead in area will be reduced even further.

Level	Power [%]	Area [%]
Async. circ.	−70	+18
Async. + Mem.	−60	+2

Table 5. Effect of asynchronous design on power and area on different levels

5. The power supply unit

Fig. 9 shows the power supply unit consisting of a rectifier and a power regulator, which are both completely analog circuits. The design of the rectifier is conventional, and f the regulator we discuss only those aspects of the behaviour that are relevant to the design of the digital circuit without going into the details of its design.

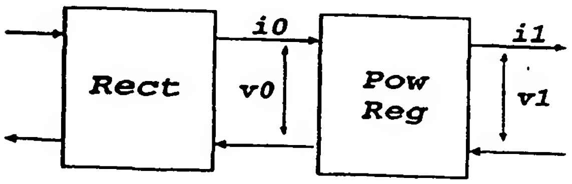

Figure 9. Power Supply Unit

To avoid interference with the communication, a power regulator has been designed that shows an almost constant load at its input. Fig. 10 shows Spice-level simulation results of such a power regulator when the input voltage V_0 is fixed at 5V. On the horizontal axis we have the activity (number of transitions per second) of the digital circuit. The input load is almost constant, since input current i_0 is almost constant over the whole range.

When the activity is low, output voltage V_1 is constant at about 3V. In this range, too much power is coming in and the regulator functions as a voltage source with output current i_1 increasing when the activity increases. The superfluous power is shunted to ground. However, i_1 reaches a *saturation point* when it reaches i_0. From this point on, no more power is shunted to ground and the regulator starts to function as a current source with output voltage V_1 decreasing when the activity increases. The regulator delivers maximum power in the middle of the range where both the outgoing voltage and the outgoing current are high. Note that these simulation results assume constant incoming RF-power. The variations in the incoming RF-power during a transaction, however, are an additional source for fluctuations in V_1, since these variations result in shifts of the saturation point.

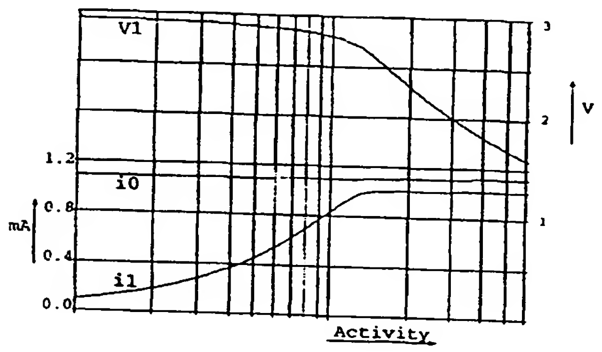

Figure 10. Power regulator behaviour

A power source with these characteristics burdens the designer of a synchronous circuit with the problem of trading off between performance and robustness. Going for maximum performance means assuming a supply voltage of 3 V in which case a transaction must be aborted if the voltage drops below 2.5 V, for instance. On the other hand, if he opts for a more robust design by choosing 2 V as the operating voltage, performance is lost when the regulator delivers 3 V. Such trade offs are not needed for an asynchronous circuit, since it always automatically gives the maximum performance for the power received.

6. Conclusions

We have designed, built and evaluated an asynchronous chip for contactless smart cards in which we have exploited the fact that asynchronous circuits:

- use little average power,
- show small current peaks, and
- operate over a wide range of the supply voltage.

Measurements and simulations showed the following advantages of this design when compared to a conventional synchronous one.

- The asynchronous circuit gives the maximum performance for the power received. This comes mainly from the fact that the asynchronous design needs less of what is the main limiting factor for the performance, namely power. Compared to a synchronous design, the asynchronous circuit needs about 40% of the power for less than 2% additional area. In addition, the automatic speed adaptation property of asynchronous circuits saves the designer from trading off between performance and robustness. Due to this property the asynchronous circuit will give free-running instead of guaranteed performance, where the difference between the two is about a factor two.
- The asynchronous design is more resilient to voltage drops, since it still operates correctly for voltages down to 1.5 V.
- The current peaks of an asynchronous circuit are less pronounced, making the requirements with respect to the buffer capacitor more modest.
- The combination of the power regulator with the asynchronous circuit gives little communication interference. In this case, the smaller current peaks and the self-adaptation property are of importance.

These advantages were so convincing that, based on the circuits presented, a product (chip for dual interface card) is being designed.

Acknowledgements

We gratefully ackowledge the other members of the Tangram team: Kees van Berkel, Marc Verra and Erwin Woutersen, and we thank Klaus Ully for helping us to get the DES converter right. This work was funded by the European Commission under Esprit TCS/ESD-LPD contract 25519 (Design Experiment DESCALE).

References

[1] E. Brunvand and R. Sproull. Translating concurrent programs into delay-insensitive circuits. In *IEEE Int. Conf. on Computer Aided Design*, pages 262–265, 1989.

[2] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International. Series in Computer Science, 1985.

[3] A. J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. *Distributed Computing*, 1(4):226–234, 1986.

[4] Identification cards - contactless integrated circuit(s) cards - proximity cards. Standard ISO/IEC 1444.

[5] National Bureau of Standards. Data encryption standard, Jan. 1977. Federal Information Processing Standards Publication 46.

[6] A. Peeters. *Single-Rail Handshake Circuits*. PhD thesis, Eindhoven University of Technology, 1996.

[7] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21:120–126, June 1978.

[8] K. van Berkel. *Handshake circuits: an asynchronous architecture for VLSI programming*. Cambridge University Press, International Series on Parallel Computation, 1993.

[9] K. van Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, and F. Schalij. Asynchronous circuits for low power: a DCC error corrector. *IEEE Design and Test*, 11(2):22–32, June 1994.

[10] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalij. The VLSI-programming language Tangram and its translation into handshake circuits. In *Proc. Eur. Conf. on Design Automation*, pages 384–389, Amsterdam, 1991.

[11] H. van Gageldonk, K. van Berkel, A. Peeters, D. Baumann, D. Gloor, and G. Stegman. An asynchronous low-power 80c51 microcontroller. In *Proc. Int. Symp. Advanced Research in Asynchronous Circuits and Systems*, pages 96–107, San Diego, 1998.